Controlling the Fidelity and Diversity of Deep Generative Models via Pseudo Density

Anonymous authors
Paper under double-blind review

Abstract

We introduce an approach to bias deep generative models, such as GANs and diffusion models, towards generating data with either enhanced fidelity or increased diversity. Our approach involves manipulating the distribution of training and generated data through a novel metric for individual samples, named pseudo density, which is based on the nearest-neighbor information from real samples. Our approach offers three distinct techniques to adjust the fidelity and diversity of deep generative models: 1) Per-sample perturbation, enabling precise adjustments for individual samples towards either more common or more unique characteristics; 2) Importance sampling during model inference to enhance either fidelity or diversity in the generated data; 3) Fine-tuning with importance sampling, which guides the generative model to learn an adjusted distribution, thus controlling fidelity and diversity. Furthermore, our fine-tuning method demonstrates the ability to improve the Frechet Inception Distance (FID) for pre-trained generative models with minimal iterations.

1 Introduction

The advent of deep generative models has revolutionized the field of image generation. Key developments marked by Variational Autoencoders (Kingma & Welling, 2013), Generative Adversarial Networks (Goodfellow et al., 2014), and the more emergent diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015), have demonstrated an unprecedented capability in producing high-quality images. They have been followed by remarkable results in applications such as super-resolution (Ledig et al., 2017; Li et al., 2022), face generation (Karras et al., 2019; 2020; 2021), and text-to-image generation (Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022). However, despite these advancements, the critical challenge of balancing fidelity (the realism of the generated images) and diversity (the variety in the generated images) remains unaddressed, despite this ability being essential for the practical applicability of these models. Our work focuses on tackling this crucial need for improved control mechanisms in generative models.

To this end, we propose to enhance control over generated images through the probability density of image data. Facing the challenge of directly estimating the density on the complex manifold of image data, we introduce a novel surrogate metric called pseudo density, which leverages the nearest-neighbor information of real samples in the feature space, utilizing an image feature extractor such as a Vision Transformer (Dosovitskiy et al., 2020). By calculating the pseudo density of the generated images, we can estimate their likelihood of appearing in the real distribution. Notably, we observe a correlation between pseudo density and the realism as well as the uniqueness of generated images. Based on the pseudo density of samples, we can modulate the probability of specific samples in both the real data and generated data. Specifically, we introduce three distinct methods to enhance the fidelity and diversity of deep generative models: 1) Per-sample perturbation, allowing precise manipulation of realism and uniqueness in any individual generated image via adversarial perturbation with pseudo density as the objective; 2) Importance sampling during model inference, enabling to enhance the fidelity or diversity in the generated data by accepting with higher or lower probability the generated samples that have higher pseudo density; 3) Fine-tuning with importance sampling, guiding the generative model to learn an adjusted distribution skewed towards data with higher or lower pseudo density, thereby controlling the fidelity and diversity.

Our extensive experiments across diverse datasets and various generative models, including different GANs and diffusion models, demonstrate the effectiveness and generality of our proposed approach in controlling the trade-off between fidelity and diversity. Additionally, our fine-tuning method can be employed to improve the Fréchet Inception Distance (FID) (Heusel et al., 2017) scores by fine-tuning pre-trained models and adjusting their fidelity-diversity trade-off. Aside from the Inception Score (IS) (Salimans et al., 2016) and the Fréchet Inception Distance (FID), as a step towards a more comprehensive evaluation, we employ precision and recall (Kynkäänniemi et al., 2019) as disentangled metrics to separately assess the fidelity and diversity of generative models.

The main contributions of this work can be summarized as follows:

1. We introduce a novel metric, termed pseudo density, for estimating the density of image data. It correlates effectively with the realism and uniqueness of individual images.
2. We propose a per-sample perturbation strategy based on pseudo density that enables adjustment of realism and uniqueness for individual samples.
3. We propose importance sampling based on pseudo density during inference, which controls the fidelity and diversity of generative models at inference time, without re-training or fine-tuning the model.
4. We propose fine-tuning with importance sampling based on pseudo density, which balances the trade-off between the fidelity and diversity of generative models without additional computational overhead during inference.
5. We demonstrate that our fine-tuning method is able to improve the Frechet Inception Distance (FID) of deep generative models by adjusting their precision (fidelity) and recall (diversity) trade-off.

2 Background and Related Work

Generative adversarial networks. GANs (Goodfellow et al., 2014) are popular generative models to learn the underlying distribution of the given training data and generate new samples. They typically consist of a generator $G(\cdot)$ parameterized by θ_g that transforms a random vector, called the latent code z, into a data sample, and a discriminator $D(\cdot)$ parameterized by θ_d that aims to distinguish the real training data from the generated samples. Both the generator and discriminator are deep neural networks that can be trained in an end-to-end manner. Therefore, training GANs involves solving a min-max optimization problem by alternatively updating the generator and the discriminator. While the original GANs may suffer from training instability, mode collapse, and poor scalability to high-resolution images, many variants (Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018; Brock et al., 2018; Karras et al., 2018; 2019) have been developed to address these problems and enable impressive high-resolution image generation.

Diffusion models. Diffusion models have emerged as a prominent class of generative models, renowned for their ability to create highly detailed and varied images, especially in a text-to-image fashion (Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022). These models, introduced in Sohl-Dickstein et al. (2015), simulate a physical diffusion process in thermodynamics: Starting with an image, noise is incrementally added over several steps until the image is transformed into pure noise. The model then learns to reverse this process, effectively reconstructing the original image from its noised state. Subsequently, the Denoising Diffusion Probabilistic Model (DDPM) (Ho et al., 2020) introduced the ϵ parameterization and adopted a new architecture based on U-Net (Ronneberger et al., 2015), obtaining significantly improved sample quality. Improved Denoising Diffusion Probabilistic Models (IDDPM) (Nichol & Dhariwal, 2021) refined the noise addition process and optimized the model's architecture to improve both the sampling efficiency and the quality of the generated images. Ablated Diffusion Models (ADM) (Dhariwal & Nichol, 2021) further improved the architecture over IDDPM and demonstrated that diffusion models can achieve image sample quality superior to that of GANs.

Fidelity and diversity of generative models. Quantitatively evaluating the performance of generative models is still an open and challenging question. Popular metrics such as the *Inception Score (IS)* (Salimans

et al., 2016) and the *Fréchet Inception Distance (FID)* (Heusel et al., 2017) are based on the overall quality (fidelity and diversity) of the generated samples. Kynkäänniemi et al. (2019) proposed the improved precision and recall metrics for generative models to separately evaluate the fidelity and the diversity. The precision is the proportion of generated samples that fall within the manifold estimated by the K nearest neighbors (KNN) of the real data, whereas the recall measures the proportion of real samples that are within the generator-induced manifold. A low precision indicates that many of the generated images do not resemble the training data, suggesting poor fidelity. By contrast, a low recall indicates a lack of diversity in the generated images. An extreme case of low recall is *mode collapse*, i.e., the failure to generate some particular categories of the multi-modal data. Several works have proposed to control the precision (fidelity) and recall (diversity) of generative models, including by truncating the latent distribution (Brock et al., 2018; Karras et al., 2019) and developing latent space samplers (Humayun et al., 2022a;b). Furthermore, discriminator rejection sampling(Azadi et al., 2018) and discriminator optimal transport (Tanaka, 2019) utilize the information retained in the GAN discriminator to improve the fidelity of the generated data during inference. However, these approaches either come with a high computational cost during the inference phase or are limited to specific GAN architectures.

Realism score and rarity score. Both the realism score (Kynkäänniemi et al., 2019) and the rarity score (Han et al., 2022), along with our new pseudo density metric, are based on the features extracted from image data using an image feature extractor. To calculate the realism score for an input image, one finds the real image sample that has the highest ratio of its k-NN radius to its distance to the input sample. The rarity score is essentially the smallest possible k-NN radius of a real sample whose k-NN hypersphere contains the input sample. The proposed pseudo density is calculated by a simple neural network that directly fits the estimated densities of real samples in the feature space. Unlike the other two metrics, our metric is specifically tailored for our control approach that manipulates data probability density. Moreover, the realism score is ineffective for samples that are rare in the real distribution, and the rarity score is not computable for outliers that are too far away from other real samples.

3 Pseudo Density

Our control approach aims to explicitly manipulate the occurring likelihood of data samples, ideally based on their probability density in the real data distribution. However, estimating the density of high-dimensional data such as images is not straightforward, primarily due to the complexities introduced by the high dimensionality and the fact that image data exist in a low-dimensional manifold. In this section, we introduce an effective metric, termed pseudo density, for estimating the density of image data. Specifically, we first employ a pre-trained image feature extractor to lower the dimensionality of the real image data while capturing the semantics. Then, for each real sample, the average distance d to its nearest neighbors is calculated. Based on the average neighbor distances, we can estimate the volume "occupied" by each sample, and the density is inversely proportional to the volume. Finally, we train a light-weight fully-connected network to fit the estimated density given the extracted feature as input. To compute the pseudo density of any generated sample, we feed it into the cascade of the feature extractor and the trained network. Note that this pseudo density metric differs from the *Fréchet Inception Distance (FID)* (Heusel et al., 2017): Pseudo density evaluates **individual** samples while the FID evaluates the **overall** performance of the generative model by calculating the distributional distance between the real and generated images. We provide more details on these steps in the following.

Estimating the real data distribution. We estimate the real data distribution in a feature space, using the extracted features of samples from the dataset. First, the features of all real samples in the dataset can be extracted by a pre-trained feature extractor, such as a Vision Transformer (ViT) (Dosovitskiy et al., 2020) trained on ImageNet (Deng et al., 2009). Then, for each data sample x_i, we find its top-k nearest neighbors based on the Euclidean distance in the feature space and compute its average distance to these k samples d_i^k. With the assumption that the sample and its neighbors are locally uniformly distributed within a sphere of radius d, we can compute the volume a sample "occupies" as $V_i^k = C \cdot (d_i^k)^n$, where n is the dimensionality of the feature space and C is a constant only conditioned on n. The density at the data point x_i is hence inversely proportional to the volume, i.e., $\hat{\rho}_i^k \propto \frac{1}{V_i^k}$. Finally, for numerical stability, we compute

Figure 1: **Images generated by StyleGAN2 (*Top*), ProjectedGAN (*Middle*), and IDDPM (*Bottom*).** The training datasets are FFHQ, LSUN-Church, and LSUN-Bedroom, respectively. For each row, the left four images obtained the highest pseudo density out of 1000 generations, and the right four obtained the lowest.

the normalized density across all data points as $\rho_i = N \cdot \hat{\rho}_i^k \cdot \frac{1}{\sum_i^N \hat{\rho}_i^k}$, where N is the total number of samples. Note that although image data (as well as their extracted features) are in a high-dimensional space, they reside in a manifold of much lower dimensionality. Hence, in practice, we set n to be a small number (e.g., 1 or 2), preventing the volume V from the curse of dimensionality.

Learning a model that predicts the density. To efficiently compute the density for any data point x and enable gradient back-propagation, we train a simple regression network $\rho(\cdot)$ to fit the density $\rho_0, \rho_1, ..., \rho_{N-1}$ with the extracted features of real samples as input $F(x_0), F(x_1, ..., F(x_{N-1})$. We observed that a basic fully-connected network consisting of only three hidden layers with fewer than one million parameters is capable of fitting well to the real data while generalizing very effectively. **We term the output of the network as pseudo density.** Compared to directly computing the density of a generated sample by injecting it into all real samples, this approach has lower computational overhead and ensures that the gradient can be back-propagated from the metric to features.

Compute pseudo density for generated samples. To calculate the pseudo density of a generated sample x, we sequentially feed it to the image feature extractor and the learned regression model $\rho(F(x))$. Intuitively, an image with a high density typically exhibits more common characteristics, whereas an image with a low density is likely to feature more unique attributes. Moreover, in the context of generated images, the lack of training on low-density data often results in reduced realism for low-density images. In Figure 1, we illustrate the effectiveness of pseudo density by showcasing generated images with either high or low pseudo density. For each model under consideration, we generate $1,000$ images, sort them according to their pseudo density, and select the ones with the highest and lowest extremes. The results evidence that the proposed metric aligns well with human perception in terms of realism and uniqueness.

4 Density-Based Perturbation

While adversarial perturbation was originally used to attack an input image to fool a classification model, their formulation applies to any other type of input to a neural network as long as an objective function is defined. In this work, we utilize pseudo density as the objective function to adversarially perturb a GAN's latent code or the input noise vector of a diffusion model, both of which we refer to as a latent vector $\boldsymbol{z}$ below.

In the context of image synthesis, such a perturbation optimizes the latent vector within its neighborhood such that it leads to a higher-density or lower-density generated image, hence increasing the realism or uniqueness of the generated image without drastically changing the image content. Let us consider a randomly-sampled latent vector $\boldsymbol{z}$. We explore the neighborhood of $\boldsymbol{z}$ using a perturbation δ based on the pseudo density. To achieve this, we employ the PGD attack (Madry et al., 2018) strategy to minimize the pseudo density of the features of $G(\boldsymbol{z}+\boldsymbol{\delta})$ under the constraint $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, where ϵ is the predefined the adversarial budget. The pseudo-code of our per-sample perturbation method is provided in Algorithm 1. In Madry et al. (2018), the perturbation is applied directly to the image and the magnitude of ϵ should be kept small to ensure that the changes are imperceptible in the image; in our context, however, the perturbation is applied to the latent code and the resulted changes in the image are intended to be noticeable. Additionally, ϵ determines the magnitude of the pseudo density shift and the extent of the content changes in the generated image. A larger ϵ yields a large change in pseudo density and also more significant changes in the content and realism/uniqueness of the generated image.

Algorithm 1 Per-Sample Perturbation Based on Pseudo Density

Require: The image generator $G(\cdot)$, the image feature extractor $F(\cdot)$, the pseudo density function $\rho(\cdot)$, the number of PGD iterations K, the step size α, the adversarial budget ϵ, and the initial latent vector $\boldsymbol{z}$.

1: Set an initial perturbation $\boldsymbol{\delta} = \mathbf{0}$
2: **for** $t = 1, ..., K$ **do**
3: Calculate the pseudo density of the perturbed sample $G(\boldsymbol{z}+\boldsymbol{\delta})$:
$f(\boldsymbol{z}, \boldsymbol{\delta}) = \rho(F(G(\boldsymbol{z}+\boldsymbol{\delta})))$
4: Update $\boldsymbol{\delta}$:
$\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \alpha \nabla_{\boldsymbol{\delta}} f(\boldsymbol{z}, \boldsymbol{\delta})$ if the goal is to increase realism
$\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} - \alpha \nabla_{\boldsymbol{\delta}} f(\boldsymbol{z}, \boldsymbol{\delta})$ if the goal is to increase uniqueness
5: $\boldsymbol{\delta} \leftarrow clip(\boldsymbol{\delta}, -\epsilon, +\epsilon)$
6: **end for**

Ensure: The perturbed latent vector $\boldsymbol{z}+\boldsymbol{\delta}$.

5 Density-Based Importance Sampling

With the help of pseudo density, not only can we perturb individual samples, but also manipulate data distributions, assigning adjusted probability to samples with different densities. To achieve this, we proceed as follows. For the given dataset, we estimate its data distribution and compute the pseudo density for all samples, as described in Section 3, and assign two different importance weights for the samples whose densities are above and below a pre-defined density threshold, respectively. Intuitively, by assigning higher weights to above-threshold or below-threshold samples for importance sampling, we can either increase or decrease the proportion of high-density images in the manipulated data distribution.

To edit the generated data distribution in a streaming fashion, one can employ rejection sampling (Azadi et al., 2018). Let the original generator's output distribution given the random latent vector be $p(x)$, and the desired distribution be $q(x)$. Furthermore, let M be the upper bound of the ratio $\frac{q(x)}{p(x)}$, i.e., $\forall x, q(x) \leq Mp(x)$. Rejection sampling proceeds by drawing a sample x^* from $p(x)$ and only accepting x^* with probability $\frac{q(x^*)}{Mp(x^*)}$. In practice, we formulate the desired distribution $q(x)$ by assigning an importance weight w to the samples that have pseudo density higher than a pre-defined threshold τ, and 1 to the other generated samples. In the case where $w > 1$, which means the high-density samples are highlighted, $q(x)$ can be obtained through iterative execution of two steps: 1) Generate a sample from the generator; 2) accept this sample if its pseudo density is higher than the threshold τ, otherwise, reject it with a probability of $1 - \frac{1}{w}$. The case where $w \leq 1$ can be handled similarly by rejection sampling.

5.1 Inference with Importance Sampling

A simple yet effective method for controlling the fidelity and diversity of generative models is to employ the above importance sampling strategy to generated images during inference, directly manipulating the

generated data distribution. To perform inference in a streaming manner, we employ rejection sampling. Different combinations of density threshold τ and importance weight w result in different effects on the data distribution, increasing either high-realism samples or high-uniqueness ones. In practice, we set the threshold to be the $\{20, 50, 80\}$ percentile of the pseudo density values of all real samples, and the weight w ranges from 0.01 to 100. Intuitively, adopting an importance weight $w > 1$ leads to more samples with a pseudo density higher than the threshold τ, hence more samples of better realism, while an importance weight $w < 1$ results in more samples with a pseudo density lower than τ, hence more variety of samples and better overall diversity. The pseudo-code of this method is provided in Algorithm 2. We demonstrate our experimental results in Section 6.

Algorithm 2 Importance Sampling during Inference

Require: The image generator $G(\cdot)$, the image feature extractor $F(\cdot)$, the pseudo density function $\rho(\cdot)$, the number of images to generate K, the density threshold τ, the importance weight w for high-density samples.

Set the number of generated images $k \leftarrow 0$.
Initialize an empty list to store generated images $I \leftarrow []$
while $k < K$ **do**
  Randomly sample a latent vector $\boldsymbol{z}$ from the prior distribution.
  Calculate the pseudo density of the generated sample $G(\boldsymbol{z})$:
    $f(\boldsymbol{z}) = \rho(F(G(\boldsymbol{z})))$
  Generate a random number $u \sim \mathcal{U}(0, 1)$
  if $w > 1$ **then**
    if $f(\boldsymbol{z}) > \tau$ or $u < \frac{1}{w}$ **then**
      Accept the generated sample $G(\boldsymbol{z})$ and push it to the list I.
      $k \leftarrow k + 1$
    end if
  else
    if $f(\boldsymbol{z}) < \tau$ or $u < w$ **then**
      Accept the generated sample $G(\boldsymbol{z})$ and push it to the list I.
      $k \leftarrow k + 1$
    end if
  end if
end while

Ensure: K generated images.

5.2 Fine-tuning with Importance Sampling

In contrast to the previous method that manipulates the generative data distribution during inference time, fine-tuning a generative model achieves it during training time. Compared to the previous method, fine-tuning does not introduce any computational overhead during inference. Furthermore, it enables the realism/uniqueness adjustment of any generated images, given their latent vectors. Similarly to the inference-time method, fine-tuning with importance sampling requires the same hyperparameters, the density threshold τ and the importance weight w, to yield different controlling effects. Before fine-tuning, we compute the pseudo density for every sample in the dataset. We can also determine the density threshold τ based on the densities of all samples. During our fine-tuning procedure, for each iteration, training examples are sampled from the dataset via the importance sampling strategy described before based on the pre-computed pseudo density.

In general, our fine-tuning approach does not modify the training algorithm except for how real samples are sampled from the dataset in each iteration. Therefore, it is compatible with any generative models including GANs and diffusion models. Moreover, it can be further improved for certain generative models such as GANs, in which we have not only the real data distribution $p_r(x)$ but also the generated data distribution $p_g(x)$ to play with during training. Specifically, when the generator and the discriminator reach

Algorithm 3 Fine-tuning GANs with Importance Sampling

Require: The batch size B, the generator $G(\cdot|\boldsymbol{\theta}_g)$, the discriminator $D(\cdot|\boldsymbol{\theta}_d)$, the number of iterations T., the density threshold τ, the importance weight w.

1: **for** $t = 1, ..., T$ **do**
2: Perform importance sampling with τ, w to obtain B real samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(B)}\}$.
3: Perform importance sampling with τ, $\frac{1}{w}$ until generating B accepted samples $\{G(\boldsymbol{z}^{(1)}), \ldots, G(\boldsymbol{z}^{(B)})\}$
4: Update D by minimizing the loss:
$-\frac{1}{B}\sum_{i=1}^{B}[D(\boldsymbol{x}^{(i)}) - D(G(\boldsymbol{z}^{(i)}))]$.
5: Perform importance sampling with τ, $\frac{1}{w}$ until generating B accepted samples $\{G(\boldsymbol{z}^{(1)}), \ldots, G(\boldsymbol{z}^{(B)})\}$
6: Update G by minimizing the loss:
$\frac{1}{B}\sum_{i=1}^{B} -D(G(\boldsymbol{z}^{(i)}))$.
7: **end for**

Ensure: The fine-tuned generator G and discriminator D

their equilibrium, we have $\frac{p_r(x)}{p_r(x)+p_g(x)} = \frac{1}{2}$ (Goodfellow et al., 2014), and thus we obtain the generated data distribution $p_g(x) = p_r(x)$. In the case where we only perform importance sampling to the real data distribution $p_r(x)$, we have $p_g(x) = Pert_r(p_r(x))$ upon convergence, where we use $Pert_g(\cdot)$ to represent the perturbation function that maps the original distribution to the sampled one. When we also apply importance sampling to the generated data distribution, the equilibrium becomes $Pert_g(p_g(x)) = Pert_r(p_r(x))$. By performing importance sampling on both $p_g(x)$ and $p_r(x)$ with different importance weights, we can achieve a stronger control over the generated data distribution $p_g(x)$. A practical choice is to set these weights as reciprocal pairs, as in our experiments. We provide the pseudo-code for the method in Algorithm 3.

Note that the proposed fine-tuning strategy can also be applied to training from scratch. However, we observed that applying fine-tuning yields comparable effectiveness but with significantly less computational overhead compared to training a model from scratch.

Additionally, our fine-tuning strategy is compatible with inference-time methods such as truncation and the proposed importance sampling strategy outlined in Section 5, enabling a combined enhancement over fidelity and diversity when applied in conjunction, as demonstrated in the Appendix A.4.

6 Experiments

In this section, we present the results of our methods on various generative models and datasets. We select Improved Denoising Diffusion Probabilistic Model (IDDPM) (Nichol & Dhariwal, 2021) in addition to Ablated Diffusion Model (ADM) (Dhariwal & Nichol, 2021), StyleGAN2 (Karras et al., 2020), StyleGAN3 (Karras et al., 2021), and ProjectedGAN (Sauer et al., 2021) to demonstrate the generality of our methods on different types of generative models. The datasets consist of LSUN-Bedroom (Yu et al., 2015), LSUN-Church (Yu et al., 2015), and FFHQ (Karras et al., 2019). The images from LSUN-Bedroom and LSUN-Church are resized to 256×256, while the FFHQ images are of resolution 1024×1024. For all experiments, we use the pre-trained checkpoints provided by the authors and use the same hyper-parameters as in the original papers for fine-tuning. More details regarding the datasets are deferred to the supplementary material, as well as more examples of generated images.

In Figure 2, we showcase examples of images generated by the pre-trained models to illustrate the effects of our proposed per-sample perturbation strategy, which achieves large density change while preserving content of the images. By applying perturbation that increases density, the generated images exhibit greater realism, characterized by simplified backgrounds, fewer objects, etc. By contrast, perturbation to lower density leads to images with higher uniqueness, featuring more complex backgrounds, the presence of rare features in the datasets, etc. We compare the effectiveness of our method with that of a random perturbation (which adds a random vector to the latent vector) in the appendix section A.2, where additional examples are provided as well.

Figure 2: **Per-sample perturbation,** applied to pre-trained models. Given random latent vectors z and their generated images (**Middle** in each panel), we perturb z to achieve higher pseudo density (**Top** in each panel) and lower pseudo density (**Bottom** in each panel). Groups from left to right, and from top to bottom: StyleGAN2 on FFHQ, ProjectedGAN on LSUN-Bedroom; StyleGAN3-T on FFHQ, IDDPM on LSUN-Bedroom; ProjectedGAN on LSUN-Church, ADM on LSUN-Bedroom.

In Figure 3, we demonstrate the trade-off between precision (fidelity) and recall (diversity) achieved by density-based importance sampling during inference. Our approach achieves a better overall precision-recall trade-off than the other two inference-time methods, polarity sampling (Humayun et al., 2022a) and truncation (Karras et al., 2019), especially in the improved-precision area. It's important to note that polarity sampling may face compatibility issues with diffusion models due to the computational complexity of the denoising process. The truncation method is adaptable to the intermediate $\mathcal{W}+$ space for StyleGAN (Karras et al., 2019; 2020; 2021) models, but is generally limited to the input $\mathcal{Z}$ space for others, which could result in suboptimal performance. For instance, for Improved DDPM, the truncation method struggles around the pre-trained trade-off point. Unlike these methods, neither the computational complexity nor particular model architectures restricts the applicability of our method since our approach directly manipulate the output data.

In Table 1, we demonstrate that our fine-tuning approach can improve the precision, recall, or FID across various datasets and generative models of different types, depending on the choices of hyperparameters, particularly the importance weight w. With larger $w > 1$, the fine-tuning procedure encourages the models to generate higher-density data, and vice versa with $w < 1$. By fine-tuning with w close to 1, the model can end up at a slightly altered trade-off point that leads to improved FID scores. Note that the density threshold τ that we use may vary for the same model. We refer the reader to the appendix for more details about the hyperparameters and fine-tuning configurations. In Figure 4, we further present examples of images generated by the pre-trained models compared with images produced by their fine-tuned versions to visually demonstrate the effects of our fine-tuning method.

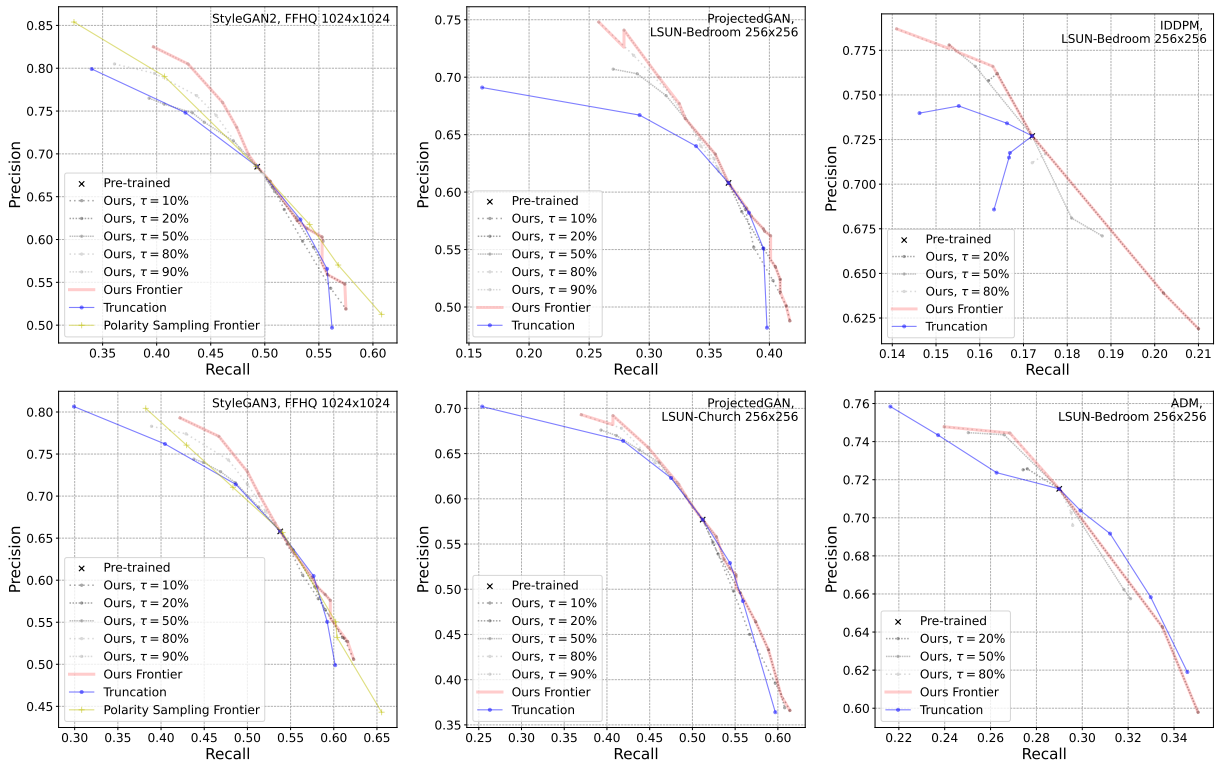

Figure 3: **Precision-recall trade-off.** For all methods, the precision and recall metrics compete with each other. For polarity sampling, we used the values reported in Humayun et al. (2022a) due to replication difficulty. For our method, we adopted different density thresholds τ for each dashed line with varying importance weights w ranging from 0.01 to 100. For better visualization, we only report part of the results for polarity sampling and truncation.

Table 1: **Improved precision/recall/FID obtained by fine-tuning with importance sampling, with different importance weights.** P stands for precision and R stands for recall. **Bold** denotes the best value in each column.

Model	P↑	R↑	FID↓	Model	P↑	R↑	FID↓
	LSUN-Bedroom 256 × 256				**FFHQ 1024 × 1024**		
Improved DDPM	0.727	0.172	10.13	StyleGAN2	0.685	0.493	2.79
+ fine-tune ($w = 33.0$)	**0.749**	0.168	9.54	+ fine-tune ($w = 33.0$)	**0.811**	0.387	36.24
+ fine-tune ($w = 0.03$)	0.648	**0.231**	13.48	+ fine-tune ($w = 0.03$)	0.558	**0.564**	10.58
+ fine-tune ($w = 2.0$)	0.737	0.175	**9.53**	+ fine-tune ($w = 0.5$)	0.675	0.500	**2.60**
ADM	0.715	0.290	6.35	StyleGAN3-T	0.658	0.538	2.81
+ fine-tune ($w = 33.0$)	**0.729**	0.284	7.70	+ fine-tune ($w = 33.0$)	**0.824**	0.382	60.89
+ fine-tune ($w = 0.03$)	0.627	**0.343**	8.15	+ fine-tune ($w = 0.03$)	0.482	**0.624**	22.34
+ fine-tune ($w = 2.0$)	0.697	0.302	**5.83**	+ fine-tune ($w = 1.5$)	0.657	0.535	**2.73**
ProjectedGAN	0.608	0.366	2.31		**LSUN-Church 256 × 256**		
+ fine-tune ($w = 33.0$)	**0.694**	0.290	3.00	ProjectedGAN	0.577	0.512	1.62
+ fine-tune ($w = 0.03$)	0.524	**0.419**	7.65	+ fine-tune ($w = 33.0$)	**0.658**	0.418	3.95
+ fine-tune ($w = 1.05$)	0.617	0.359	**2.22**	+ fine-tune ($w = 0.03$)	0.494	**0.540**	7.47
				+ fine-tune ($w = 2.0$)	0.586	0.507	**1.58**

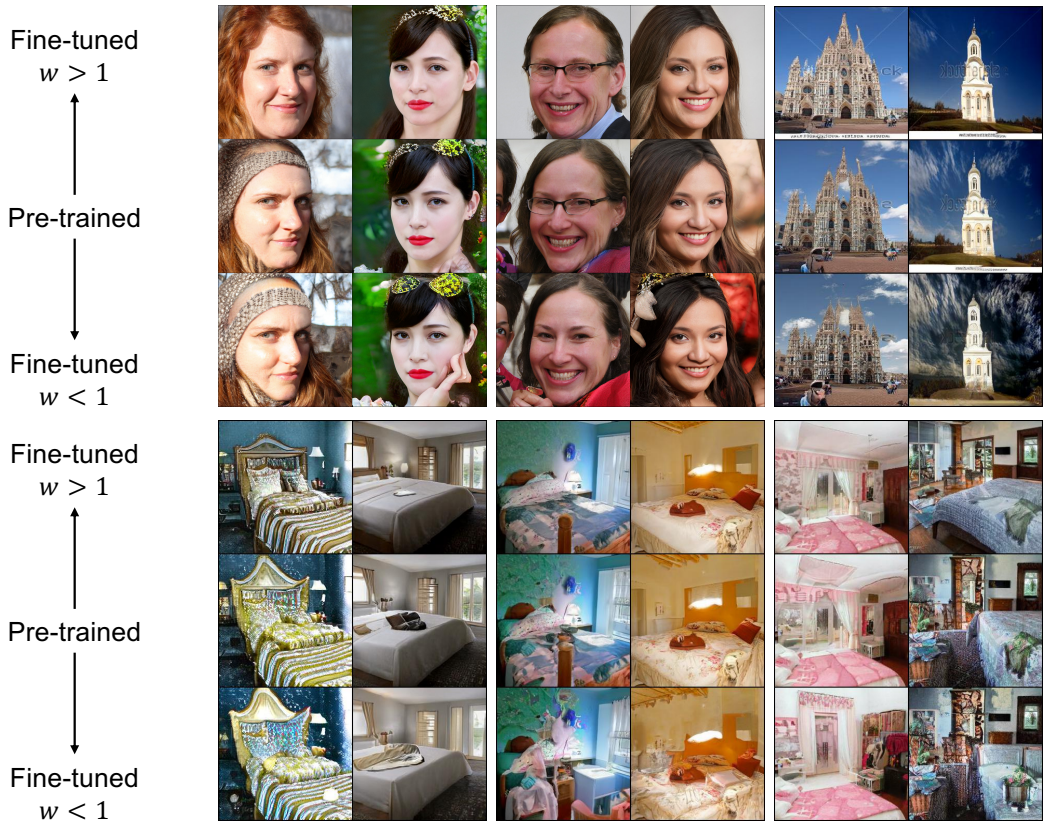

Figure 4: **Images generated by pre-trained and fine-tuned models.** For each column in a panel, the same latent vector is used by the pre-trained model and the fine-tuned versions for image generation. Fine-tuning with $w > 1$ biases the generative model to output more high-density data. Groups from left to right, and from top to bottom: StyleGAN2 on FFHQ, ProjectedGAN on LSUN-Bedroom; StyleGAN3-T on FFHQ, IDDPM on LSUN-Bedroom; ProjectedGAN on LSUN-Church, ADM on LSUN-Bedroom.

7 Conclusion

In this study, we have introduced a simple and effective approach for estimating the density of image data, enabling us to devise inference-time methods and a fine-tuning strategy for biasing deep generative models into outputting data with either higher fidelity or higher diversity. We have shown that our method can greatly improve the fidelity or diversity in the generated data without significantly altering their primary subject and structure, making it of great interest to applications such as image editing. We have also demonstrated that adjusting the balance between precision (fidelity) and recall (diversity) through fine-tuning can improve the Frechet Inception Distance (FID) for models pre-trained in a standard manner. This underscores the importance of considering both fidelity and diversity in the evaluation of generative models instead of relying solely on FID as a performance metric. Future research may aim to evaluate and improve various density-based sampling strategies for optimized their efficacy. Another potential research interest lies in adapting the proposed control approach to conditional generation tasks such as text-to-image synthesis.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223. PMLR, 2017.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, 30, 2017.

Jiyeon Han, Hwanil Choi, Yunjey Choi, Junho Kim, Jung-Woo Ha, and Jaesik Choi. Rarity score: A new metric to evaluate the uncommonness of synthesized images. *arXiv preprint arXiv:2206.08549*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10641–10650, June 2022a.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Magnet: Uniform sampling from deep generative network manifolds without retraining. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=r5qumLiYwf9`.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34: 852–863, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, 2017.

Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International conference on learning representations*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29, 2016.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Akinori Tanaka. Discriminator optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. Logan: Latent optimisation for generative adversarial networks. *arXiv preprint arXiv:1912.00953*, 2019.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

A Appendix

A.1 Training Details, Evaluation Details, and Hyper-parameters

Datasets The FFHQ (Flickr-Faces-HQ) dataset (Karras et al., 2019) is a high-quality collection of human face images consisting of 70k images with the resolution of 1024×1024. The LSUN-Church and LSUN-Bedroom are subsets of the LSUN (Large-scale Scene Understanding) dataset (Yu et al., 2015). For LSUN-Church, we use all 126k images in the dataset and resize them to 256×256 resolution. For LSUN-Bedroom, we sample the first 200k images from the dataset and apply center crop, resizing them to 256×256 resolution as well. We followed the same pre-processing procedures as in previous works.

Training details We follow the same training hyper-parameters published in the original papers, except the number of GPUs, while keeping the same batch size per GPU. For GANs models in our experiments, we used 2 GPUs, and a single GPU for diffusion models. For StyleGAN2/3 models, we fine-tuned for, measured in thousands of real images fed to the discriminator, 80 thousand images. For Projected GAN, we fine-tuned for 40 thousand images. To improve FIDs, we only fine-tuned all GANs models for 12 thousand images. As for diffusion models in our experiments, we fine-tuned them for $128 \times 20000 = 2560$k images.

Regarding the computation of pseudo density, we use $n = 1$ for all datasets and $K = 10$ for all except LSUN-Bedroom where we set $K = 20$.

Evaluation details To evaluate the FID, precision, and recall for a generative model, we use the model to generate $50,000$ images every time and then calculate the metrics against the training dataset. For diffusion models, however, we only generate $10,0000$ images every time for computational efficiency. In addition, we use the uniform stride from DDIM (Song et al., 2020) and each sampling process takes 50 steps.

Per-sample perturbation hyper-parameters We employed PGD attack (Madry et al., 2018) whose hyper-parameters involve the number of steps K, the step size α, and the adversarial budget ϵ. For all GANs models in our experiments, we adopted $K = 10, \alpha = 0.025$, and $\epsilon = 0.1$. For diffusion models, we adopted $K = 5, \alpha = 0.0025$, and $\epsilon = 0.0125$.

Density-based importance sampling hyper-parameters The two relevant hyper-parameters are the density threshold τ and the importance weight w of above-threshold samples. Their optimal values may vary across different datasets and models. We conducted a sweep for τ with values from $\{20, 50, 80\}$ percentiles of the estimated densities of real images. We also performed sweeps for w with values from $\{0.01, 0.03, 0.1, 10, 33.0, 100\}$. We show the optimal values found in Table 2. When aiming to improve FIDs, we kept a density threshold of 50 percentile and performed importance sampling only on the real data. We conducted a sweep for w from $\{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 1.05, 1.1, 1.3, 1.5, 1.7, 2.0\}$ for GANs models and only $\{0.5, 2.0\}$ for diffusion models.

Table 2: **Optimal hyper-parameters for importance sampling.** Optimal hyper-parameters vary with different goals, which are improving precision (Precision↗), improving recall (Recall↗), and improving FID (FID↘). In the first two scenarios, the selection was not based on maximizing the precision/recall gain but on the (subjective) optimal trade-off.

	Precision↗		Recall↗		FID↘	
Model, Dataset	τ	w	τ	w	τ	w
StyleGAN2, FFHQ	80%	33.0	20%	0.03	50%	0.5
StyleGAN3-T, FFHQ	80%	33.0	20%	0.03	50%	1.5
ProjectedGAN, LSUN-Church	50%	33.0	50%	0.03	50%	2.0
ProjectedGAN, LSUN-Bedroom	50%	33.0	50%	0.03	50%	1.05
IDDPM, LSUN-Bedroom	80%	33.0	20%	0.03	50%	2.0
ADM, LSUN-Bedroom	80%	33.0	20%	0.03	50%	2.0

A.2 Discriminator Output versus Pseudo Density

Many works, including LOGAN (Wu et al., 2019), discriminator rejection sampling (Azadi et al., 2018), and discriminator optimal transport (Tanaka, 2019), have explored the output of a trained discriminator as a metric for sample realism. Ideally, discriminator output may indicate sample realism in the context of Generative Adversarial Networks since the discriminator learns to tell if a sample is real or generated. Azadi et al. (2018) demonstrated that discriminator output aligns well with realism in a 2D toy dataset. However, our experiments suggested that it works less ideally for image data, which are in much higher dimensions. In Figure 5, we showcased the results of per-sample perturbation based on pseudo density and discriminator output. Additionally, we compared to random perturbation, which replaces the gradient-descent direction with a random direction while keeping the same magnitude in each step of the PGD attack.

A.3 Extra Computational Overhead

The proposed rejection sampling strategies, as described in Section 5, Algorithm 2 and Algorithm 3, result in extra computational cost since only a proportion of the generated samples are accepted. Given the importance weight $w < 1$ and let the density threshold be $p\%$ of estimated densities of the data, assuming the generated data perfectly fit the real data, a generated sample is accepted with probability $0.01p + (1 - 0.01p) \times w$ on average, hence the computational cost is approximately $\frac{1}{0.01p+(1-0.01p)\times w}$ times as before. The case where $w > 1$ can be handled similarly. In our experiments, generating samples via StyleGAN2 with $p = 50, w = 0.01$ slows down from the original 13 sec/kimgs to 26 sec/kimgs while generating with $p = 20, w = 0.01$ slows down to 60 sec/kimgs. Fine-tuning StyleGAN2 with $p = 80, w = 33.0$ drops from 56 sec/kimgs to 74 sec/kimgs. However, fine-tuning diffusion models does not suffer from obvious slower training speed since it only involves importance sampling on the real data where rejection sampling is not applied.

A.4 Combination of Fine-tuning and Inference-Time Methods

Not only the pre-trained models but also the models fine-tuned by our methods can be readily combined with any inference-time methods such as our proposed importance sampling strategy and truncation (Karras et al., 2019) to further boost the precision/recall improvement. In Table 3, we demonstrate that applying them to fine-tuned models can achieve even higher precision/recall than either fine-tuning or solely applying inference-time methods.

Table 3: **Applying truncation or our importance sampling strategy during inference further improves the precision/recall.** P stands for precision and R stands for recall. **Bold** denotes the best value in each column.

Model	P↑	R↑
StyleGAN2	0.685	0.493
+ Fine-tune ($\tau = 80\%, w = 33.0$)	0.811	0.387
+ Truncation ($\Phi = 0.7$)	0.854	0.233
+ Sampling ($\tau = 90\%, w = 100$)	0.825	0.397
+ Fine-tune ($\tau = 80\%, w = 33.0$) + Truncation ($\Phi = 0.7$)	**0.938**	0.163
+ Fine-tune ($\tau = 80\%, w = 33.0$) + Sampling ($\tau = 90\%, w = 100$)	0.872	0.198
+ Fine-tune ($\tau = 20\%, w = 0.03$)	0.558	0.564
+ Truncation ($\Phi = 1.3$)	0.491	0.571
+ Sampling ($\tau = 20\%, w = 0.01$)	0.548	0.574
+ Fine-tune ($\tau = 20\%, w = 0.03$) + Truncation ($\Phi = 1.3$)	0.386	**0.617**
+ Fine-tune ($\tau = 20\%, w = 0.03$) + Sampling ($\tau = 20\%, w = 0.01$)	0.473	0.597

A.5 Additional Qualitative Result

In Figures 6, 7, 8 9, 10, 11, we present images generated by per-sample perturbation, compared to the original generated images. In Figures 12, 13, 14, 15, 16, 17, we present images generated by pre-trained models and fine-tuned ones.

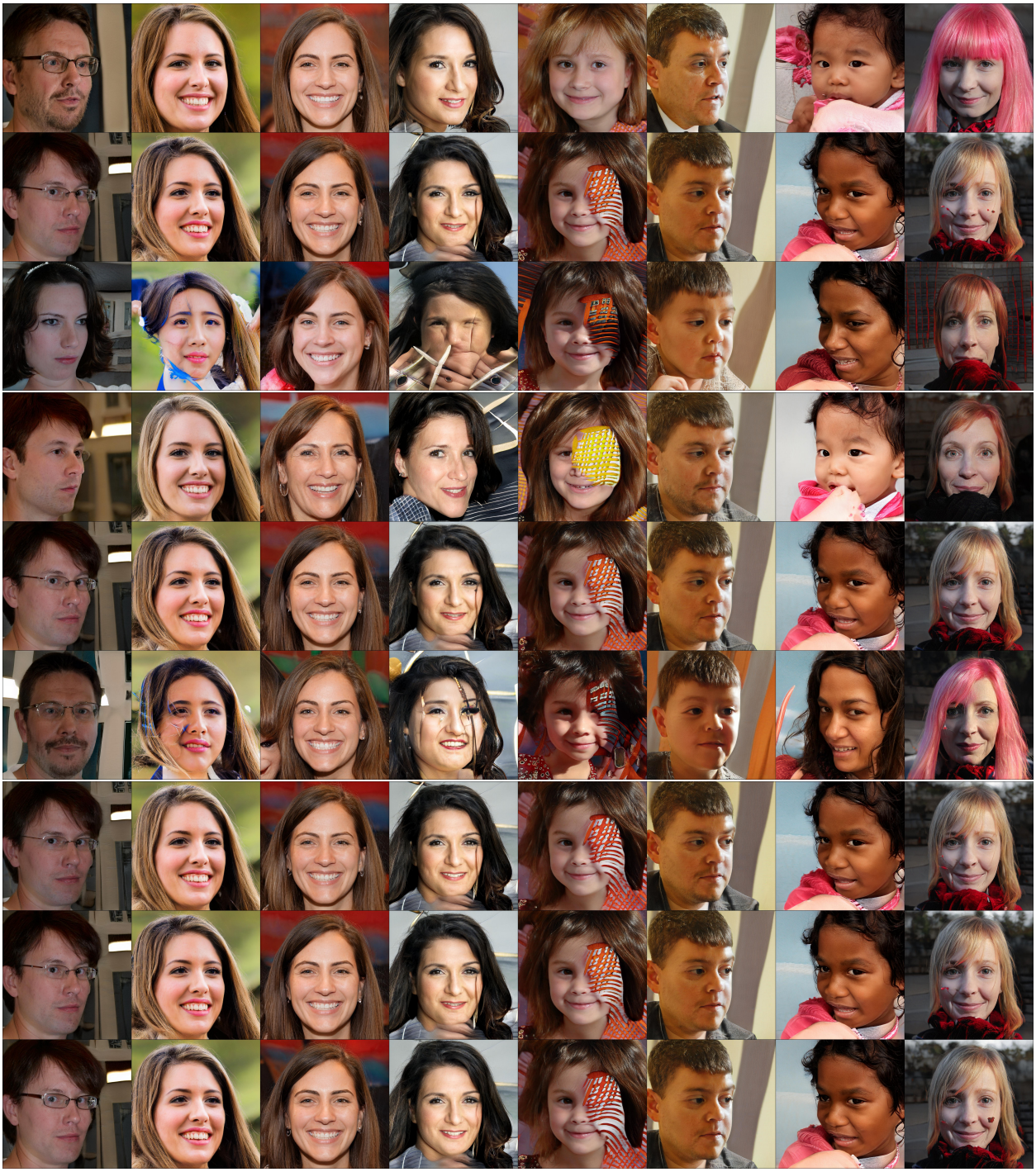

Figure 5: **Images generated by StyleGAN2 pre-trained on FFHQ with per-sample perturbation.** (**Top panel**): Pseudo Density as the objective for PGD. (**Middle panel**): Discriminator output as the objective for PGD. (**Bottom panel**): Random perturbation. In each panel, the middle row displays images generated by the pre-trained model, and the top row corresponds to images perturbed for better realism while the images in the bottom row are perturbed for better diversity.

Figure 6: **Images generated by StyleGAN2 pre-trained on FFHQ with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

Figure 7: **Images generated by StyleGAN3-T pre-trained on FFHQ with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

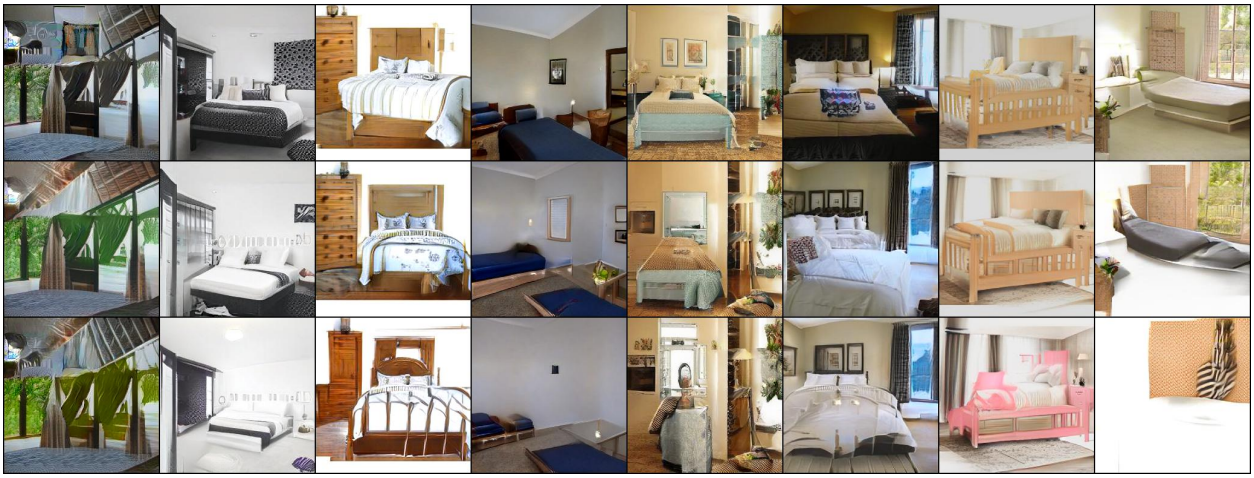

Figure 8: **Images generated by ProjectedGAN pre-trained on LSUN-Bedroom with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

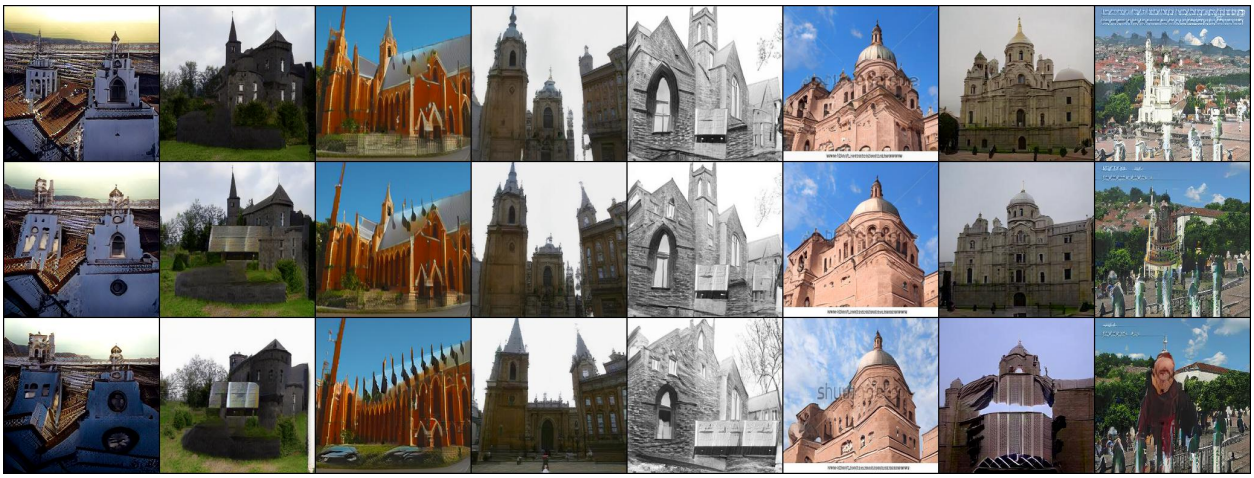

Figure 9: **Images generated by ProjectedGAN pre-trained on LSUN-Church with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

Figure 10: **Images generated by IDDPM pre-trained on LSUN-Bedroom with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

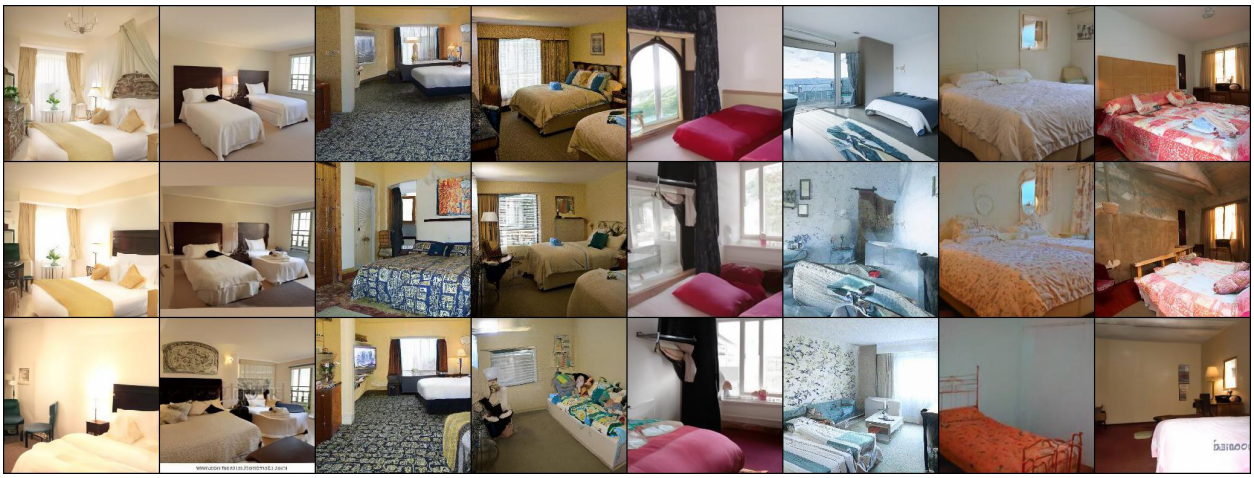

Figure 11: **Images generated by ADM pre-trained on LSUN-Bedroom with per-sample perturbation.** (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the same model with perturbed latent vectors for better realism. (**Bottom**): Generated by the same model with perturbed latent vectors for better diversity.

Figure 12: **Images generated by StyleGAN2 trained on FFHQ after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.

Figure 13: **Images generated by StyleGAN3-T trained on FFHQ after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.

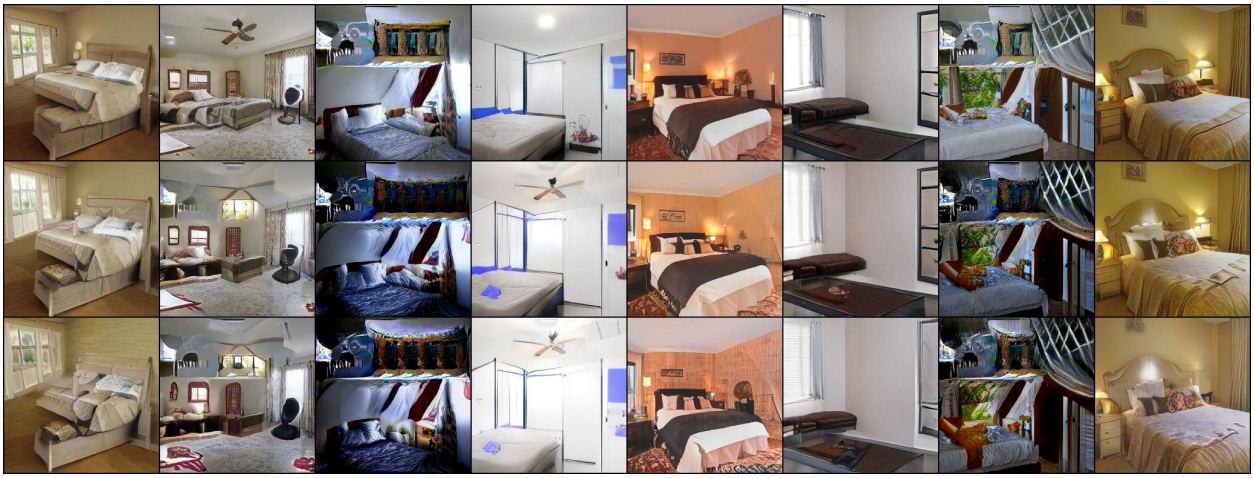

Figure 14: **Images generated by ProjectedGAN trained on LSUN-Bedroom after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.

Figure 15: **Images generated by ProjectedGAN trained on LSUN-Church after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.

Figure 16: **Images generated by IDDPM trained on LSUN-Bedroom after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.

Figure 17: **Images generated by ADM trained on LSUN-Bedroom after fine-tuning with importance sampling.** Within each column, images are generated with the same latent code. (**Middle**): Images generated by the pre-trained model. (**Top**): Generated by the model fine-tuned for better fidelity. (**Bottom**): Generated by the model fine-tuned for better diversity.